1. given x, y, z, penalty.
2. update x so that it is feasible for bounds and any linear constraints.
3. evaluate f, c, g, J and compute norm_c, max_infeas_bar, merit
4. iterate = 0, make sure blow-up is not signified
5. **BEGIN (main loop)**
6. compute residuals C_RES_l, C_RES_u, A_RES_l, A_RES_u, X_RES_l, X_RES_u.
7. IF iterate = 0 then check_optimal goto 9.
8. if x_k has moved then
   if sqp used
   if ~bfgs
   then compute J^ya from sqp
   end
   compute At ya from sqp and then check optimal
   update primal_vl, dual_vl, comp_vl and Y if necessary
   end
   if ~bfgs
   compute Jty from cauchy
   end
   compute Aty from cauchy and then check optimal
   update primal_vl, dual_vl, comp_vl and Y if necessary
   else
   if sqp computed
   compute Jty and Aty and then check optimal
   update primal_vl, dual_vl, comp_vl and Y if necessary
   end
   compute Aty and Jty from cauchy and then check optimal
   update primal_vl, dual_vl, comp_vl and Y if necessary
   end
9. If optimal or iterate = max_iterate, go to 13.
10. check subproblem optimal based on point accepted in 8, which is same as in nlp%Y
11. determine whether to increase penalty parameter (if blowup, subproblem optimal, other?)
12. evaluate merit function if penalty parameter has changed
13. print : iterate, primal_vl, dual_vl, comp_vl, penalty parameter, merit)
14. if optimal or iterate = max_iterate, EXIT
15. iterate = iterate + 1
16. IF BFGS then update B(s,_f or s_c, gradLx, gradLxnew, B) note: make sure update knows when it is first iterate so that it can handle first iterate.
17. compute predictor step/mults
18. Evaluate H possibly at Cauchy mults.
19. compute cauchy step/mults and model decrease
20. possibly compute sqp step/sqp mults and model decrease at full step
21. IF sqp not computed OR sqp computed but not good then eval. fnew, cnew at Cauchy pt.
    ELSE evaluate fnew, cnew at sqp step.
22. IF blow-up, then signify.
23. compute ratio
24. print : (on separate line) everything to do with predictor, cauchy, and sqp : TR
25. IF “blow up” then decrease trust-region
    ELSE if success, then
    1. update trust-region

2. compute gradLx(g,J,Y) where Y corresponds to whether cauchy or sqp step was used
3. x <- xnew (depends on weather SQP was used), c <- cnew, f <- new, norm_c = norm_cnew
4. update merit
5. evaluate g <- gnew, J <- Jnew
6. compute gradLxnext(g,J,Y) where is same Y from 24.2.

ELSE failure, then update trust-region.

26. END (main loop)